Asterisk and Obelisk: Motion Codes for Passive Tagging

Aakar Gupta
University of Waterloo
Waterloo, Canada
aakar.gupta@uwaterloo.ca

Jiushan Yang
University of Toronto
Toronto, Canada
jiushan.yang@mail.utoronto.ca

Ravin Balakrishnan
University of Toronto
Toronto, Canada
ravin@dgp.toronto.edu

ABSTRACT

Machine readable passive tags for tagging physical objects are ubiquitous today. We propose Motion Codes, a passive tagging mechanism that is based on the kinesthetic motion of the user's hand. Here, the tag comprises of a visual pattern that is displayed on a physical surface. To scan the tag and receive the encoded information, the user simply traces their finger over the pattern. The user wears an inertial motion sensing (IMU) ring on the finger that records the traced pattern. We design two motion code schemes, Asterisk and Obelisk that rely on directional vector data processed from the IMU. We evaluate both schemes for the effects of orientation, size, and data density on their accuracies. We further conduct an in-depth analysis of the sources of motion deviations in the ring data as compared to the ground truth finger movement data. Overall, Asterisk achieves a 95% accuracy for an information capacity of 16.8 million possible sequences.

Author Keywords

Motion Code, Machine Readable Tags, Ring, QR Code

INTRODUCTION

Machine readable tags such as QR codes, magnet stripes, and RFID tags lend a digital identity to physical objects or serve as physical hyperlinks to digital data. They are immensely useful for commercial applications for tracking and rapid identification, as well as for consumer use via smartphone apps. We propose an alternative tagging scheme called *motioncodes* that is based on kinesthetic motion of the user's hand. The user traces their finger on a pattern (a *motiontag*) displayed on a physical surface to get the encoded information. The motion tag is a passive visual pattern that can simply be printed out on paper and affixed onto any surface. The user wears an inertial motion sensing unit (IMU) that records the traced pattern.

Motion codes' IMU-based scanning enables low-power, low-cost, lightweight scanning that can be easily incorporated into miniature wearable devices such as smartwatches and smartrings. Optical scanning of visual markers using cameras requires higher power and bulkier hardware. Further, motion tags are passive, inexpensive to mass-produce, and can be

UIST '18, October 14–17, 2018, Berlin, Germany

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-5948-1/18/10…$15.00

DOI: https://doi.org/10.1145/3242587.3242637

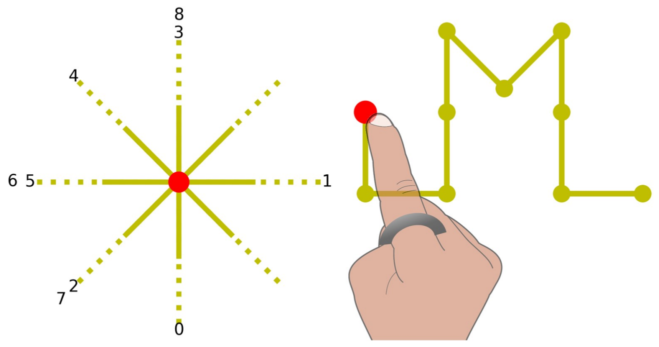


Figure 1. User traces the finger on a motion code while wearing an IMU ring to get encoded data. Figure shows our two proposed schemes: (left) Asterisk: User starts at center and traces outward & inward in the numbered order. (right) Obelisk: User traces the path. (Basis Angle: 45°)

easily augmented on to various surfaces. In this paper, we use a finger-worn IMU ring to scan the motion tag. Scanning of machine readable tags adds to the repertoire of tasks that multipurpose smartrings are increasingly becoming popular for, including notifications, authentication, and fitness tracking.

We propose two motion code schemes - Asterisk and Obelisk. The two schemes demonstrate the trade-off between accuracy and speed: Asterisk is more accurate while Obelisk is faster to perform. We evaluate both schemes for the effects of orientation, size, and data density on their accuracies. Overall, the Asterisk coding scheme with an information capacity of 16.8 million unique sequences achieved a 95% accuracy with a scanning time of 10s. The Asterisk scheme with 100k unique sequences achieved 96% accuracy with a scanning time of 7s. The Obelisk scheme with 12.5k unique sequences achieved a 95% accuracy with a scanning time of 5.5s.

The primary contribution of this paper is that it is the first exploration of motion-based passive tagging. The sub-contributions are - (a) the design and implementation of two motion code schemes; (b) the evaluation of their performance which shows that motion codes are feasible and useful with high accuracy and capacity; (c) an analysis of motion code accuracy that uncovers the sources of error, and; (d) example scenarios and extensions for motion codes.

RELATED WORK

Physical Tags for Identification

Physical tagging techniques can consist of active or passive tags. While active tags have their own battery and a transmitter, passive tags have no battery. Passive RFID, magnetic stripe, or NFC [3] are popular passive tags differing in their scanning range. However, these tagging mechanisms have higher installation costs and cannot be generated and distributed electronically. Optical tags (like 1D barcodes, 2D QR codes) are passive tags that do not have these limitations, can be simply printed out and are consequently in popular use. Another optical approach is using invisible ink for tags visible only under UV light [14]. Recently, techniques to embed 3D printed objects with unobtrusive optical tags readable using computational [16] or terahertz imaging [29] have been proposed. However, these have special installation and scanning requirements. Acoustic barcodes [13] propose patterns of physical notches engraved into a surface, which when swiped using a fingernail produces a sound that can be reduced to a unique acoustic ID. This is close to our work in that it requires user interaction with the code. However, etching notches into physical objects again leads to installation overheads. Further, they require the user to have long nails or to retrieve a pen.

Finger-worn Sensors

Existing works have put cameras [6, 18], magnets [12, 7, 4], vibration motors [11], infrared reflectors [10, 9, 21], touch sensors [25, 24], and inertial motion units (IMUs) [15, 19] on the finger. These sensors have been used for detecting finger gestures on 2D surfaces [15, 31], in air [19, 12], on other fingers [7, 6], or on the ring itself [24]. Magic Finger [31] and 3DTouch [19] place optical flow sensors (like the ones in optical mice) on the finger-tip to track finger displacement on a surface. LightRing [15] overcomes the finger-tip placement and places it on the proximal phalange closer to the real-world use. However, it is limited in that it can only perform local tracking of the finger within a 3-5cm circle. Since IMUs cannot measure displacement accurately, they have mostly been used for detecting orientation or coarse gestures such as jerks, taps, swipes [2, 30] or walking steps [22]. For motion code detection, we use the directional velocity derived from a 9 degrees-of-freedom (dof) IMU.

Motion Menus and Motion Correlation

Motion gestures have mostly been used with external tracking of hands and fingers. Internal motion tracking has been used for enabling motion marking menus [20, 5] that utilize tilt. Oakley et al [20] use 3 pitch tilts of a smartphone for a 4-item menu. Bailley et al [5] show accurate working of 7 roll angles with a 2-level menu for 49 items (sequences). Motion correlation is a related technique where users match their motion with that of an onscreen object to select it. Most work in this space focuses on matching gaze motion with the object motion's shape or rhythm [28, 8, 26] or on matching arm motion detected by computer vision for a limited number of distinct motion patterns (10-20). Technically, WaveTrace [27] is closest to our work, using a smartwatch IMU's yaw-pitch data for correlating user's hand movement with the on-screen rotating object's x-y coordinates for up to 8 distinct motion patterns. Motion codes use a finger-worn IMU's yaw-pitch-roll and directional motion data for detecting a distinct tracing motion for a static pattern where distinct motion patterns are in the order of millions. To our best knowledge, this is the first ever investigation into motion based tagging.

MOTION CODES

Motion codes simply encode a particular piece of information which is sent in form of the IMU readings by the user to the ring when they trace a pattern. The basic usage scenario for a motion code is this: The user is wearing an IMU ring on the index finger. The user spots a motion code pattern on a physical surface, walks up to it and traces the pattern with their index finger. The ring decodes the motion data into the associated information and then (or directly) transmits the information to a smartphone or the cloud where the user can access it later. While in our current implementation, users wear rings, motion codes could possible be made to work with other devices such as smartwatches.

Desired Usability & Functional Properties

Motion codes should satisfy the following properties to ease usability: 1) The motion code pattern should span a small area to avoid large hand movements that are physically demanding. 2) Even though placing the ring at the tip will make the detection a lot easier, the ring should be worn at the proximal phalange, to resemble daily use. Motion codes should satisfy the following functional properties to ensure its utility: 1) The encoding should focus on *maximizing information capacity* for the smallest durations. 2) Decoding should be *independent of finger or ring orientation*, working for scenarios where the finger is at different angles from the surface or at different angles with respect to the real world, and the ring is at any rotated position on the finger. 3) Assuming an evenly planar surface, decoding should be *independent of surface orientation* and be consistent across vertical and horizontal surfaces. 4) *False Positives:* Since fingers move randomly all the time, the ring needs to know when the user intends to scan an actual pattern. We propose two motion code schemes - *Asterisk* and *Obelisk*, that offer different usability tradeoffs.

Design Approach & Hardware

We designate a hard physical button on the ring that the user presses with the thumb of the same hand just before and after drawing the pattern to indicate detection start/stop. For the pattern, we initially sought to have freeform 2D curves. However, computing vector displacement from acceleration and orientation data is highly error-prone. We use a discrete approach where the patterns consist of a series of linear strokes. The first stroke is considered the base stroke and the angles between the subsequent strokes and the first stroke encode the information. We detail this process in the next section.

To build the ring (Figure 2), we used BNO055 [1], a 9 dof sensor which was mounted on an Arduino Pro Micro and attached to a velcro ring with the physical button. We use the open source BNO055 library [23] to get the smoothed linear acceleration ($acceleration - gravity$) and orientation (yaw, pitch, roll) of the ring.

ASTERISK

Figure 3 shows a sample Asterisk motion code pattern. To scan an Asterisk pattern: (1) the user places the index finger at the center spot, (2) clicks the ring button to start tracking, (3) then runs the finger on the line (performs a *stroke*) labeled 0, (4) pauses momentarily at the end of the line's solid part,

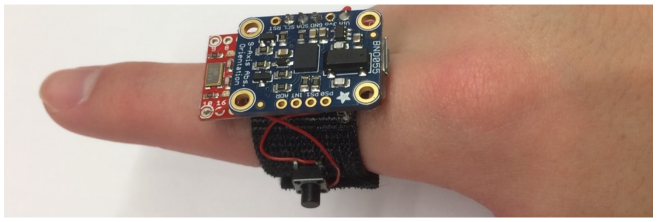


Figure 2. The 9dof motion sensing ring. The user clicks the button to start scanning a code, traces the pattern, and clicks again to stop.

(5) strokes inwards back to the center, (6) momentarily pauses again and repeats 3-6 for every line following the numbering order. Upon returning to the center after *line* 8, the user clicks the button to stop tracking, and lifts off the finger.

Encoding

In a simple barcode, the bar width is the *basis* of information with $b = 2$ possible widths. For an encoding with n bars (*size*), the number of possible sequences are 2^n. In Asterisk, the *basis* is the angle of a line relative to the first line (*line* 0). Figure 3 shows the encoding with $b = 8$ possible values: 0°, 45°, 90°...315° in the sequence with $n = 8$ digits (8 lines after the first one). For instance, *line* 1 can have 8 possible angles relative to *line* 0 , so will *line* 2, and so on. Notice that the encoding allows for overlapping lines. This results in an information capacity of 8^8 = 16.8 million possible sequences (24 bits). Figure 3 shows a pattern corresponding to one of those sequences and the expected pairs of stroke vectors to be decoded from the motion data. We fixed *line* 0 to always be a downward vertical line to make the starting stroke symmetric for both left and right-handed users.

Since the minimum relative angle is 45°, the margin of error for a stroke is within 22.5° on either side. For instance, if the angle of the stroke for *line* 1 ($\vec{s_1}$) is decoded to be a value >=67.5° and <112.5, then it is considered as 90°. We call this as the *basis angle* which forms the $b = 360/(basis\ angle)$ possible sequences. Thus, lower the basis angle, higher is the number of possible sequences for the same size n. However, this would also mean a lower margin of error, requiring the decoding to be more accurate.

User Guidelines

There are two sources of error which the decoding software cannot control: (1) the IMU's internal noise, and (2) the user's stroking errors. Because users can trace the pattern with varying velocities, varying pauses, and varying finger and arm joint motion over the course of a pattern, it is difficult to handle every variation. We therefore prescribe a simple stroking protocol with two guidelines for the user to follow when tracing out the pattern: (1) The user should move in quick bursts of motion from one vertex to the next punctuated by momentary pauses at each vertex without lifting the finger up. We refer to this motion between two vertices as a *stroke*. (2) The user should avoid using the finger joints for movement and use the wrist, elbow and shoulder joints instead. Since the ring is at the proximal phalange, finger movements which involve the two distal phalanges and no (or very little) movement of the proximal one cannot possibly be detected by the ring and is therefore limited. The guidelines are not overtly strict and allows for variations in the quickness of movement, duration of pauses, and finger movements.

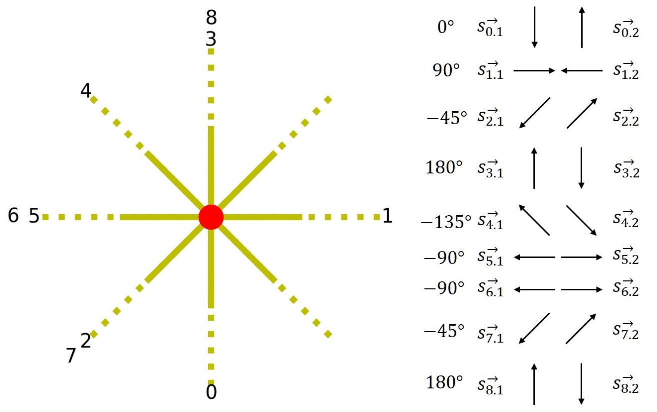

8
3
4
6 5
1
7 2
0
0° $\vec{s_{0.1}}$ $\vec{s_{0.2}}$
90° $\vec{s_{1.1}}$ $\vec{s_{1.2}}$
−45° $\vec{s_{2.1}}$ $\vec{s_{2.2}}$
180° $\vec{s_{3.1}}$ $\vec{s_{3.2}}$
−135° $\vec{s_{4.1}}$ $\vec{s_{4.2}}$
−90° $\vec{s_{5.1}}$ $\vec{s_{5.2}}$
−90° $\vec{s_{6.1}}$ $\vec{s_{6.2}}$
−45° $\vec{s_{7.1}}$ $\vec{s_{7.2}}$
180° $\vec{s_{8.1}}$ $\vec{s_{8.2}}$

Figure 3. (left) An asterisk pattern with basis angle 45° and size $n = 8$. (right) The expected sequence of stroke direction vectors for this pattern.

Preliminary Usability Evaluation

We conducted a preliminary evaluation with 3 participants (1 female, age 17-20) where they traced 10 patterns on paper. We found that they were easily able to understand and follow the guidelines. Participants' hands sometimes occluded the number labels which led them to pause longer and twist their fingers to peek at the numbers. We therefore added the dotted segments to each line (Figure 3). This ensured less occluded number labels while keeping the trace length same.

Decoding

We now describe our algorithm to decode motion data between the two button clicks into a pattern. The algorithm assumes that it is decoding an Asterisk pattern with a predefined basis angle and size. The inclusion of a pattern's scheme, basis angle and size inside the pattern itself is a subject for future work. The motion data between the two clicks consists of a stream of linear acceleration and orientation values sampled at 100Hz. Our goal is to compute the direction vectors for all $n+1$ strokes and then calculate their relative angle with respect to the first stroke vector. For this, we first need to segment the data into $n+1$ strokes. As a first step, the algorithm discards all patterns with durations $> 2(n+1)$ or $< (n+1)/3$ seconds. Figure 4 shows a block diagram of the Asterisk decoding algorithm. Let's take a step-by-step look.

World Acceleration (Figure 5a): The relative angle to first stroke ensures that the algorithm is independent of the initial finger or ring orientation and independent of surface orientation as long the physical surface is flat. However, finger or ring orientation may change while tracing. We make the decoding independent of finger or ring orientation changes by converting the accelerometer's linear acceleration into absolute acceleration with respect to the real world (*world acceleration*) using the yaw, pitch, and roll. Next, we apply a low-pass filter to filter out the high frequency noise (Figure 5b).

Velocity: Next, we compute velocity (Figure 5c) and pass it through a high-pass filter to remove the drift Figure 5d.

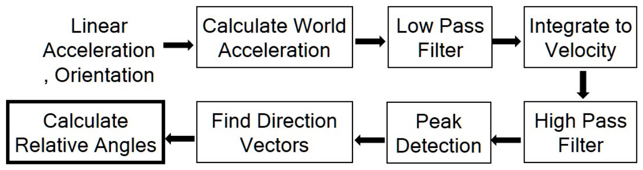


Figure 4. Decoding algorithm block diagram for Asterisk

Peak Detection: As a succession of quick movements and pauses, strokes are characterized as peaks in the velocity magnitude stream as seen in Figure 5e. We run a peak detector to detect $n+1$ peak-pairs that correspond to the $n+1$ line pairs (Figure 5f). The algorithm identifies values that are greater than the 10 nearby values within 100ms on either side. This weeds out peaks from jerks in finger motion during a single stroke. Assuming that the quick movements of a user are performed with reasonably consistent speeds across all lines, the detector removes any peaks that are less than 10dB of the maximum peak velocity to remove any spurious peaks due to low velocity noisy data when the finger is paused. We now detect the outward-inward peak-pairs: starting from the first two peaks, if the distance between them is <2s and if the angle between them is 180° with a total error margin of 30°, they marked as the first peak-pair and the process repeats for the next two. If not, the second and the third peak are tested similarly and so on until the end. At the end, if there are $<n+1$ peak pairs, the pattern is discarded as corrupt. If there are $>n+1$ peak-pairs, the top $n+1$ by magnitude are picked.

Stroke Direction Vectors: The peak is the point of maximum velocity of a stroke. A stroke's direction, however, will be defined by the complete movement from one vertex to the next. To approximate this, we take an average of k velocity vectors at either side of a peak (including the peak). This k should be large enough for estimating the stroke's direction, but small enough to not exceed the range of that stroke. After conducting experiments with varying speeds, this was fixed at $k = 7$. Thus we have $n+1$ stroke direction vector pairs $\vec{s_{0.1}}, \vec{s_{0.2}}, \vec{s_{1.1}}, \vec{s_{1.2}} \ldots \vec{s_{n.1}}, \vec{s_{n.2}}$.

Relative Angles: The decoder calculates the relative angle between strokes $\vec{s_0}$ and $\vec{s_i}$ by averaging the relative angles of outward and inward strokes: angle between $\vec{s_{0.1}}$ & $\vec{s_{1.1}}$ *and* between $\vec{s_{0.2}}$ & $\vec{s_{1.2}}$ (computed using dot product). This minimizes the effect of a deviated outward or inward finger movement. All n relative angles are rounded to the nearest possibility (e.g. for a 45° basis angle, a 68° relative angle is rounded off to 90°).

Our algorithm minimizes the following errors : (1) accidental button clicks & incomplete trials, (2) ring or finger orientation changes without displacement, (3) variations in stroke velocities & pause durations, (4) accidental jerks, (5) small deviations in outward or inward movement.

OBELISK

There are two primary issues with Asterisk - (1) Getting a single angle requires two lines, out and back in. (2) As evident in the usability evaluation, the occluded numbering caused problems for the users and reduced the speed. We designed Obelisk to tackle these two issues.

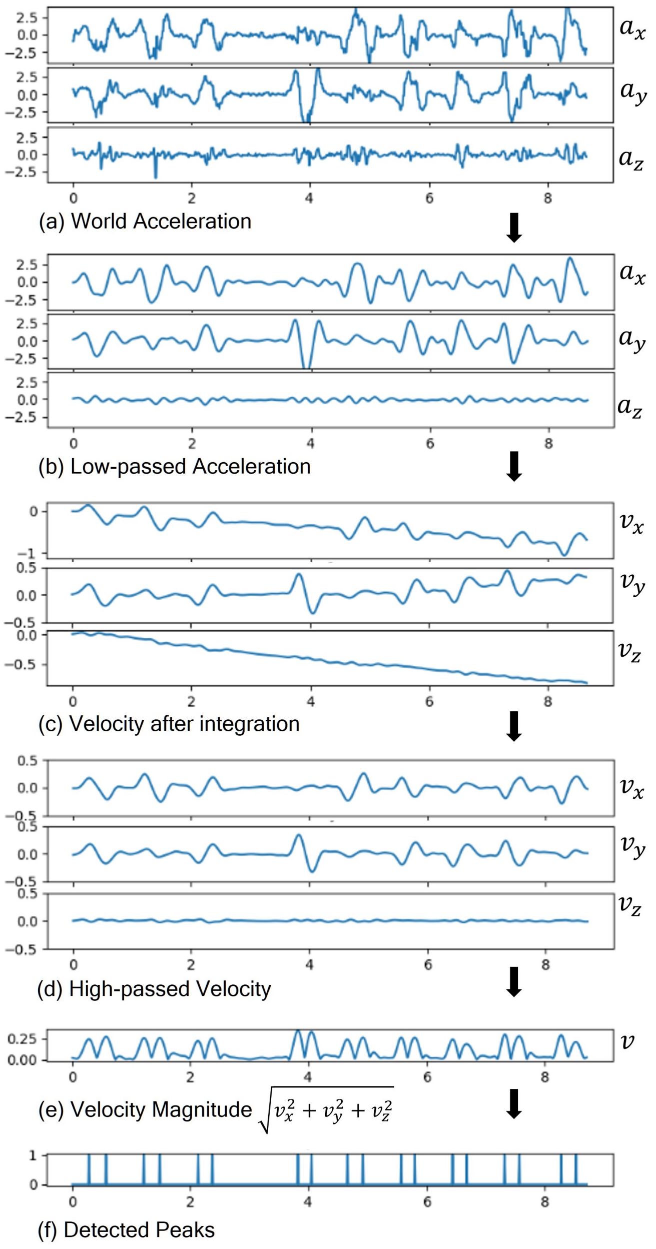


Figure 5. An example of velocity peak detection for an Asterisk pattern.

Figure 1(right) shows a sample Obelisk motion code pattern. The user starts by (1) placing the index finger at the red spot, (2) clicks the button to start tracking, (3) then strokes along the first line, (4) pauses momentarily at the circle (vertex), (5) then repeats 3-4 for every line until the end, (6) clicks the button to stop tracking, and lifts off the finger. Obelisk allows the user to simply follow a single path thus precluding the need for double movements and numbering. To make sure that the user does not face a situation where two lines emerge from a single vertex, we discard all such cases from the encoding, as well as cases when a vertex falls on a previous line.

Encoding

In Obelisk, the basis is again the angle of a line relative to the first line. However, since Obelisk does not allow coinciding vertices, a 180° angle relative to the previous line, is not a possibility and so base for Obelisk is $b = (360/(basis\ angle)) - 1$. Thus, for a basis angle of 45°, we have 7 possibilities. Figure 1 shows an example with $n = 8$. Note that for the k^{th} line, if an angle leads to the next vertex coinciding with a prior line, that possibility is also discarded. However, intersections between two lines without the coinciding vertex is allowed. Upon running a 100k random generation session, 75% of the sequences satisfied this criteria and so the number of possible sequences is $.75b^n$= approx. 4.3 million for $n = 8$, $b = 7$.

Decoding

The Obelisk decoder follows a similar approach to Asterisk. For peak detection, after removing low-velocity peaks, it picks the top $n+1$ peaks. For relative angle, it measures the angle between a stroke and the first stroke. The relative angle in Obelisk can also be measured with respect to the previous stroke as opposed to the first stroke of a pattern. This will remove the dependency of every stroke's accuracy exclusively on the first stroke. Here, a stroke's accuracy will be dependent partially on all prior lines instead of just the first one. While this may result in a worse performance, it could be beneficial in cases where the user drifts away more and more every subsequent stroke, possibly trying to rush through the pattern. The same measurement can also be applied to Asterisk, but makes less sense there. We explore this in our experiment.

EVALUATION

We conducted a study to evaluate the accuracies under different conditions. Aside from the SCHEME (Asterisk and Obelisk), we considered 3 independent variables that could influence both IMU and user error: (1) BASIS ANGLE (or ANGLE): A lower basis angle increases the number of possible sequences, but also leaves a lower margin of error. We selected three basis angles for evaluation: 45°, 36°, 30° resulting in bases b of 8, 10, 12 respectively. (2) LINE LENGTH (or LENGTH): Tracing a line with a longer length would take more time and therefore, a shorter length for each line is desirable. The question is, is there a lower limit beyond which a decrement in length affects the performance? We therefore evaluated two line lengths: $3cm$ and $1.5cm$. Given that average finger width lies around 1cm, $1.5cm$ is an effective test for lower limit. For Asterisk, a lower length might also lead to higher occlusion affecting speed & accuracy. (3) ORIENTATION: We evaluate the codes on both *horizontal (H)* and *vertical (V)* surfaces to test if our algorithm is agnostic of orientation. Thus, a total of $2x3x2x2 =$ 24 conditions were evaluated. We fixed the size $n = 8$ (so a total of 9 lines including *line* 0) which allowed for a large number of possible sequences within reasonable durations. This also allows us to study performances for smaller sizes by limiting the analysis to $< n$ lines.

Our evaluation intended to investigate the following metrics: (1) *Pattern Accuracy*: The average % of patterns that were performed completely correctly. We evaluate how pattern accuracy varies under the effects of our independent variables. (2) *Pattern Duration*: The average pattern tracing time from the starting button click until the stopping button click. (3) *Stroke Deviation*: The deviation of the stroke and its comparison with the ground truth touch data. (4) *Error Detection & Correction*.

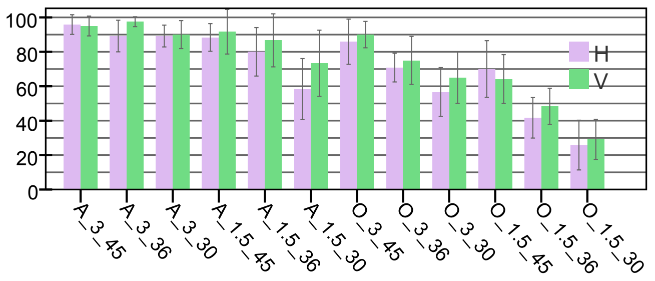


Figure 6. Mean pattern accuracy % for Horizontal & Vertical orientation for all conditions. A_3_45- Asterisk 3cm 45°. Error Bars 95% CI.

Further, we investigated the possibility of potentially inconsistent performance between patterns of the same encoding scheme because strokes pertaining to a certain angle (wrt the prior stroke) may be error prone (for instance participants may neglect pausing for 0° angles wrt the previous line in Obelisk). We investigate this issue in our evaluation. For each condition, a participant performed 10 randomly generated motion code patterns (trials) for a total of 240 trials per participant. However, if a randomly generated $3cm$ Obelisk pattern's vertical span exceeded beyond the 19cm tablet touchscreen length, it was replaced with another pattern.

Experiment Design

The experiment followed a within-subjects design for all 4 factors. SCHEME and ANGLE were fully counterbalanced, and ORIENTATION and LENGTH were partially counterbalanced (2 permutations: H1.5, H3, V1.5, V3 and V3, V1.5, H3, H1.5) to result in a Latin square design of size 12 (2 SCHEMES *x* 3 ANGLES *x* 2 ORIENTATION-LENGTHS).

The user wears the ring and performs the gestures on a touch tablet which is on a horizontal table or fixed vertically against a wall. Participants sat during the whole study. The height of the tablet for the vertical orientation was adjusted depending on the participant's sitting height. While the gestures could have been performed on any surface, the tablet allows us to get the ground truth touch data so that we can distinguish between user errors and algorithmic errors. The touchscreen also consisted of 2 buttons: *Next* and *Discard*.

12 participants (5 female, 7 male, 2 left-handed, age 22-49, $\mu = 29.3$) took part in the study. Left-handed participants wore the ring in their left index finger. Initially, the participants were explained the concept and the motion code scheme for the condition they were going to perform first. They were then asked to trace the pattern according to the guidelines. After clicking the button at the end, they pressed *Next* to move to the next trial. If they thought they made a mistake, they were asked to press the *Discard* button and redo the trial. The participants performed five practice trials and were corrected upon any procedural errors they made. They went through a similar explanation and practice before the first time they performed the other motion scheme. They were given a 1min break after every 20 trials and a 5min break whenever the next condition was for a different scheme from the previous. The study lasted 90mins. Overall, we had 2 SCHEMES *x* 3 ANGLES *x* 2 LENGTHS *x* 2 ORIENTATIONS *x* 10 *trials x* 12 *users* = 2880 *trials*.

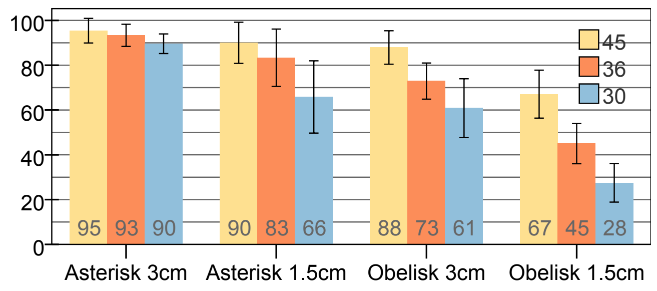


Figure 7. Mean pattern accuracy % per SCHEME per LENGTH per ANGLE. No. at bottom of the bar is accuracy %. Error Bars: 95% CI.

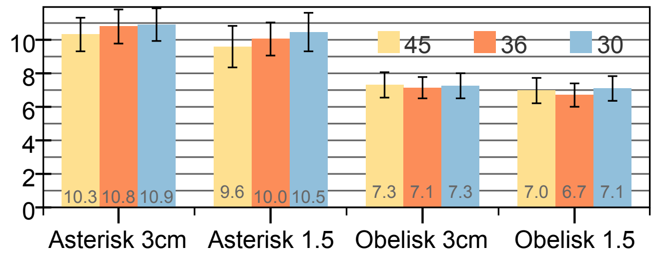


Figure 8. Mean pattern duration (s) % per SCHEME per LENGTH per ANGLE. Error Bars: 95% CI.

RESULTS

In all, 109 of 2880 trials (3.8%) were discarded and redone (around 9 of 240 per participant). Most of these errors were a result of participants doing trials one after the other - for instance, pressing *Next* without clicking the ring button or due to finger slipping while trying to trace a pattern hurriedly.

Pattern Accuracy

We conducted a four-way repeated measures ANOVA on the pattern accuracy and found main effects for SCHEME, LENGTH, and ANGLE. We also found 2-way interaction effects of SCHEME*LENGTH, SCHEME*ANGLE, and LENGTH*ANGLE. No 3-way or 4-way interaction effects were found. Importantly, no main effect or interaction effect involving ORIENTATION was found. Figure 6 shows that the horizontal and vertical accuracies for all 12 condition are very similar. Similar results were obtained for time which we discuss later. Thus, both *Asterisk and Obelisk can be used in either orientation without loss of accuracy or speed*. Logically, both schemes should also work for slanted surfaces, but this needs to be confirmed in an evaluation.

Because of the 2-way interactions, we computed simple main effects for SCHEME, LENGTH, & ANGLE. Table 1 shows the statistical test results for significant main & interaction effects, and for the simple main effects. Figure 7 shows the mean pattern accuracy % per SCHEME per LENGTH per ANGLE. The accuracy for Asterisk 3cm 45° is 95% which is highly encouraging. Plus, Asterisk 3cm's accuracy remains >90% even at 36° & 30°. (Table 1: ANGLE A 3cm) shows that these differences are not significant.) Thus *Asterisk 3cm enables a much higher information density* (12^8) *without a major loss of accuracy*. 1.5cm 45° accuracy lowers statistically, but remains at 90%. For 36° & 30°, however, Asterisk dips significantly and fails to maintain its performance. Obelisk's accuracy is

Pattern Accuracy Effect	F value	p value	η^2
SCHEME	$F(1,11) = 64.003$	**<0.001**	0.853
LENGTH	$F(1,11) = 65.886$	**<0.001**	0.857
ANGLE	$F(1,11) = 67.982$	**<0.001**	0.861
SCHEME*LENGTH	$F(1,11) = 11.159$	**<0.01**	0.504
SCHEME*ANGLE	$F(1.3,13.9) = 12.851$	**<0.005**	0.539
LENGTH*ANGLE	$F(2,22) = 9.934$	**<0.005**	0.475
SCHEME 3cm 45°	$F(1,11) = 4.304$	>0.05	0.281
SCHEME 3cm 36°	$F(1,11) = 66.863$	**<0.001**	0.859
SCHEME 3cm 30°	$F(1,11) = 22.701$	**<0.005**	0.674
SCHEME 1.5cm 45°	$F(1,11) = 22.198$	**<0.005**	0.669
SCHEME 1.5cm 36°	$F(1,11) = 43.917$	**<0.001**	0.8
SCHEME 1.5cm 30°	$F(1,11) = 33.636$	**<0.001**	0.754
LENGTH A 45°	$F(1,11) = 5.755$	**<0.05**	0.343
LENGTH A 36°	$F(1,11) = 3.940$	>0.073	0.264
LENGTH A 30°	$F(1,11) = 13.048$	**<0.005**	0.543
LENGTH O 45°	$F(1,11) = 28.061$	**<0.001**	0.718
LENGTH O 36°	$F(1,11) = 80.817$	**<0.001**	0.88
LENGTH O 30°	$F(1,11) = 38.938$	**<0.001**	0.78
ANGLE A 3cm	$F(2,22) = 15.553$	>0.05	0.203
Pairwise	45:36, 45:30, 36:30	>.05,>.05,>.05	
ANGLE A 1.5cm	$F(2,22) = 15.553$	**<0.001**	0.586
Pairwise	45:36, 45:30, 36:30	>.05,**<.005,<.005**	
ANGLE O 3cm	$F(2,22) = 18.945$	**<0.001**	0.633
Pairwise	45:36, 45:30, 36:30	**<.01,<.005**,>.05	
ANGLE O 1.5cm	$F(2,22) = 45.026$	**<0.001**	0.804
Pairwise	45:36, 45:30, 36:30	**<.005,<.001,<.005**	

Table 1. Statistical test results (Main, interaction, and simple main effects) for Pattern Accuracy. A-Asterisk, O-Obelisk. SCHEME*ANGLE is with Greenhouse-Geisser correction since sphericity was not satisfied.

generally lower than Asterisk. For 3cm 45°, it is at 88% and not significantly different (Table 1: SCHEME 3cm 45°). But, it dips significantly for 36° & 30°. The accuracies plummet for 1.5cm. Thus, considering raw accuracy without error correction, the promising patterns are Asterisk 3cm 45°, 36°, 30°, and to a lesser extent Asterisk 1.5cm 45° and Obelisk 3cm 45°. We also computed accuracies using the previous stroke instead of the first stroke and found no significant differences and very similar mean values for all conditions.

Pattern Duration

We again conducted a four-way repeated measures ANOVA on the pattern duration and found main effects for SCHEME, and ANGLE, and an interaction effect of SCHEME*ANGLE. No other effects were found. Because of the 2-way interaction, we computed simple main effects for SCHEME & ANGLE. Table 2 shows the statistical test results. The mean durations for Asterisk 3cm are 10.3s, 10.8, and 10.9s (Figure 8). This implies that every stroke took half a second on average. Expectedly, Obelisk takes significantly less time (approx. 7s) than Asterisk for all angles. Surprisingly, length did not affect the durations for either Asterisk or Obelisk. This means that 1.5cm is worse on accuracy and does not do better on speed either and can be safely rejected as an option. Participants reported that for 1.5cm, the movement was more controlled to avoid overshooting into the next line. Table 2 shows that while angle had a main effect, the pairwise comparisons are mostly not significant. This implies that the angle could potentially be reduced even further without affecting the duration. All participants, barring one, reported that they preferred Obelisk. One participant said that he liked Asterisk since all motion was in a small, fixed region.

Pattern Duration Effect	F value	p value	η^2
SCHEME	$F(1,11) = 43.437$	**<0.001**	0.798
ANGLE	$F(1,11) = 3.619$	**<0.05**	0.248
SCHEME*ANGLE	$F(2,22) = 8.100$	**<0.005**	0.424
SCHEME 45°	$F(1,11) = 33.117$	**<0.001**	0.751
SCHEME 36°	$F(1,11) = 53.337$	**<0.001**	0.829
SCHEME 30°	$F(1,11) = 39.637$	**<0.001**	0.783
ANGLE Asterisk	$F(2,22) = 6.149$	**<0.01**	0.359
Pairwise	45:36, 45:30, 36:30	>.05,**<.05**,>.05	
ANGLE Obelisk	$F(2,22) = 2.170$	>0.05	0.165
Pairwise	45:36, 45:30, 36:30	>.05,>.05,>.05	

Table 2. Statistical test results for Pattern Duration.

The accuracies slightly dipped with every additional line (Figure 9) and the time taken increases by approx. 0.5s for Obelisk and by 1s for Asterisk. Thus, coding schemes with a smaller number of lines (n) will have better accuracy and lower time. Asterisk 45° maintains an accuracy $> 95\%$ till line 8, enabling a capacity of 8^8=16.8 million unique sequences (24 bits) while taking 10.3s. Asterisk 36°'s accuracy falls below 95% beyond line 5 and enables a capacity of $10^5 = 100k$ unique sequences (16.6 bits) while taking 7.5s. Obelisk 45° is more suited for applications that require lower capacities and faster scanning time, enabling a 95% accuracy for a sequence of size 5 with 12k capacity (13.6 bits) in 5.5s.

To investigate the possibility of strokes of certain angles (wrt the previous stroke) being more inaccurate than others, we analyzed the stroke accuracy by angle. We found no significant or visible differences in any of the 12 conditions. Thus, all patterns within a specific encoding have consistent performance regardless of the angles contained.

DISCUSSION OF PATTERN ACCURACY

We now do an in-depth discussion to understand the sources of error and their alleviation.

Overshooting & False Starts

Overshooting a vertex seems to be a primary cause for the observed differences in conditions. After overshooting, the user simply moves from this overshot position to the next vertex in a straight line. This corrective movement results in a deviated vector. For the same overshot position, the straight line to the next vertex will result in a considerably higher deviation for the 1.5cm LENGTH, compared to 3cm (Figure 10(left)). This causes the drop in accuracy. Overshooting also causes the differences in Asterisk and Obelisk. In Asterisk, overshooting in the outward direction does not cause any deviation, only in the inward. And thus the out,in average reduces this deviation. In Obelisk, overshooting is an issue for every stroke causing higher deviation and reduced accuracy. A related issue present only for the first stroke, but which affects all subsequent strokes is the *false start*: if the user starts at a point slightly displaced from the exact center, then the stroke to the next vertex will be deviated from line 0, causing a deviation in the first line. Again, due to the same reasons as overshooting, a false start results in more deviation for 1.5cm and Obelisk.

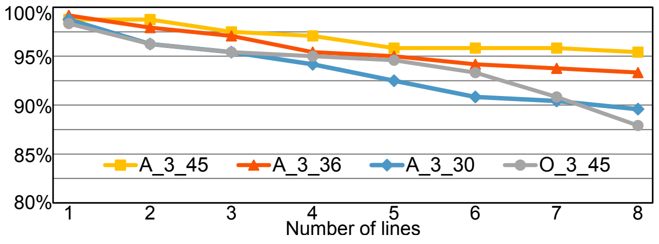
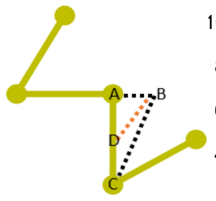


Figure 9. Accuracy at the n^{th} line for all Asterisk 3cm conditions and Obelisk 3cm 45° condition. (The other conditions are not illustrated due to their low accuracies). This shows that encodings with smaller sizes will have higher accuracies. (Note that the vertical axis starts at 80%).



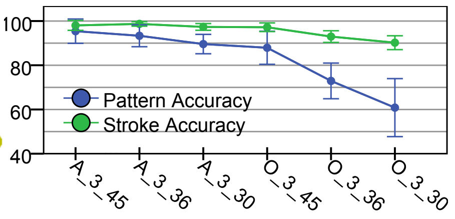


Figure 10. (left) Illustrates the higher deviation angle in 1.5cm because of overshooting. User overshoots to B, then tries to go to the next vertex. $\vec{BD}$ to $\vec{AD}$ has a larger deviation than $\vec{BC}$ to $\vec{AC}$. (right) Stroke level accuracy % for 3cm conditions compared to pattern-level accuracy. Error Bars: 95% CI.

This hypothesis is confirmed by looking at the touch data from the touchscreen. Figure 11 shows the mean deviation in degrees of the user's strokes for touch and motion data. The orange line at the bottom shows the *TouchVertices* deviation in degrees. This is calculated using only the points where the user paused on the touchscreen after a quick stroke. For example, if user pauses at P1, then strokes and pauses at P2, the TouchVertices direction vector is P2-P1 which is then used to get the TouchVertices deviation. This directly reflects overshooting. As the figure shows, the TouchVertices deviation increases from 3cm to 1.5cm for both Asterisk and Obelisk. Similarly, Obelisk 3cm shows higher TouchVertices deviation than Asterisk 3cm and same for the 1.5cm conditions.

This deviation is mirrored in the StrokeVelocity deviation (blue line at the top), which is the stroke velocity vector's deviation from its expected angle with respect to the first stroke velocity vector. However, how did the StrokeVelocity deviation become so large compared to TouchVertices? This is because the user's stroke not only deviates at the vertices but also in the middle. The stroke velocity vector is calculated around its peak which lies somewhere in the middle. To confirm this, we look at TouchStroke deviation which is calculated by averaging the deviations of all the points of a stroke on the touchscreen between (and including) the two vertices. TouchStroke deviation (green line in the middle) is much closer to StrokeVelocity deviation and explains the gap.

However, after the Asterisk 3cm conditions, StrokeVelocity deviation moves away when the overshooting deviation happens in 1.5cm or Obelisk. This increase in StrokeVelocity deviation slope compared to Touch slopes (from O_3_30 to O_1.5_45, for example) is because while StrokeVelocity deviation for every stroke also incorporates deviations due to false starts, the

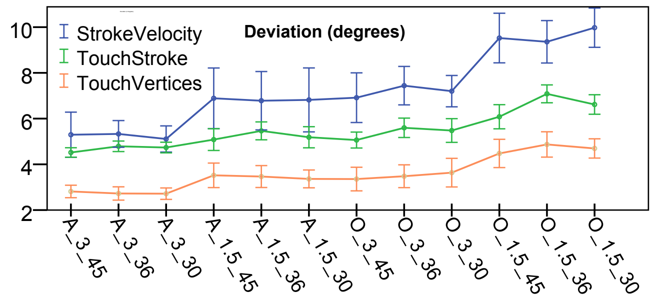


Figure 11. Mean deviation in degrees of the user's strokes for all conditions. StrokeVelocity: Deviation of a stroke's relative angle from the line's real angle. TouchStroke: Total Deviation of the stroke vector calculated from all the touch points from one vertex to the next, from the real line. TouchVertices: Deviation of the stroke vector calculated from the touch points only at the vertices of a line, from the real line. Error Bars: 95% CI.

touch deviations do not as they are calculated relative to the fixed line on the touchscreen and not the user's touch points of that line. Since false starts also affect 1.5cm and Obelisk more, it translates to the StrokeVelocity deviation line moving away. A second reason is that participants were more prone to distal-phalange finger movement in 1.5cm (since it's a smaller distance) and in Obelisk. This was of course captured in the touch data, but not in the motion data.

However, the proximity in deviation values for Asterisk 3cm conditions show that *in the absence of overshooting, false starts, and distal phalange movements, our decoding algorithm reproduces the user's movement directions almost perfectly* and there are no hidden factors that are not addressed or not explained. This modeling of the deviation performance has some optimistic implications - (1) *Longer lengths will have even better accuracy.* And so long as a length increment does not significantly increase duration, both Asterisk and Obelisk will benefit from it. (2) *A motion coding scheme that minimizes the effects of overshooting and false starts will have a better accuracy.*

Stroke Accuracy

The deviation increments due to overshooting and false starts, however, do not explain the exponential plummet in pattern accuracy. This is because, it only takes a single incorrect relative angle to render the whole pattern incorrect. Figure 10(right) shows the mean stroke accuracy (% of strokes correctly performed) for the 3cm conditions. We see that a linear decrease in the stroke accuracy triggers an exponential decrease in pattern accuracy. For Obelisk 3cm 30°, a 90% stroke accuracy translates to a 61% pattern accuracy. Thus, the majority of dips in accuracy are explained by magnification of small errors to big errors: small false starts and overshooting, result in higher deviations, higher deviations result in occasional strokes' relative angles to exceed their margin of error, and a single such exceedance in a pattern results in an incorrect pattern, rendering other correct strokes useless. While longer length and better motion coding scheme designs will attack the roots of the problem, the stroke accuracy graph also implies that detection and correction of single or double stroke errors will also impact pattern accuracy.

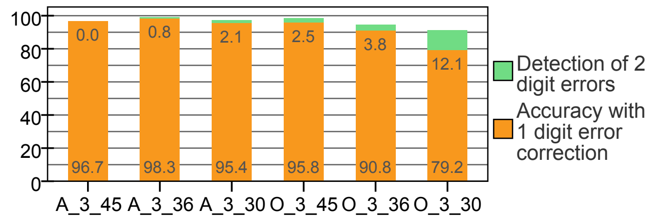


Figure 12. The mean pattern accuracy % after Hamming(8,4) correction of 1-digit error (in orange), and the additional % of 2-digit detected errors (in green at the top)

Error Detection & Correction

We now look at how error detection and correction can improve the reliability of Asterisk and Obelisk. This involves using some digits of a message as error-correcting check digits, and the rest as data digits. We calculate the Hamming distance between the decoded octal (base 8, 45°), decimal (base 10, 36°), and duodecimal (base 12, 30°) digits and the corresponding expected sequence. This is essentially the number of erroneous strokes (or digits) in a pattern. According to the extended Hamming code (8, 4) [17], for an 8-digit message (like ours), 1-digit error correction and 2-digit error detection is possible using 3 check digits and four data digits.

Figure 12 shows (in orange) the mean pattern accuracy % upon correction of a single stroke error for the 3cm conditions. All Asterisk conditions and Obelisk 45° now have a >95% pattern accuracy. Further, the figure shows the % of 2-digit errors that can be additionally detected (but not corrected). Thus, Obelisk 36° has 90.8% accuracy and a further 3.8% error detection, making the undetected errors <5%.

However, this does mean that the code's information density will reduce exponentially. E.g., Asterisk 30° reduces to ($12^4 = 20736$) sequences (14.3 bits). Since Asterisk 3cm's accuracy and duration were not significantly affected by lowering basis angles, angles<30° may increase the information density of 1-digit error corrected Asterisk codes and merit investigation.

If we focus only on 1-digit error detection, we only need a single parity digit. This implies that Asterisk 36° will enable 10^7 =10 million unique sequences (23.3 bits) with a <2% undetected error rate (93% accuracy+5.3%detected errors). Similarly, Obelisk 45° will enable $0.75 * 9^7$ =823k sequences (19.7 bits) with a <5% undetected error rate. Real-time error detection can be performed to inform users (via ring vibrations or audio) so that they can repeat the trace.

IN-AIR EVALUATION

So far, we have discussed touch-tracing of motion codes. We now explore the possibility of in-air scanning of motion codes. Users may want to scan a motion code from a distance. Further, physically touching a surface might not be preferred by users in certain contexts. In our preliminary explorations, in-air tracing proved to be highly imprecise for 36° and 30°. We conducted a small evaluation for 45° Asterisk and Obelisk with the 3cm stroke length. The patterns were displayed on a vertical touchscreen at the same height as earlier and the participants were seated 50cm away from it. Participants were not allowed to use any armrest. The schemes were counterbalanced. For each scheme, participants performed 20 randomly generated patterns in air. One change that was made to the Obelisk pattern was that if a randomly generated pattern's onscreen length on either of the left or right side of the origin exceeded 9cm, that pattern was replaced with another. This was done so that the user does not stretch the hand too far out of their comfort zone. The rest of the procedure including the decoding algorithm was similar to the earlier study. 8 participants (2 female, all right-handed, age 22-30, $\mu = 27.5$), all different from prior study took part. The study lasted 20mins. Overall, we had 2 SCHEMES $x\ 20\ trials\ x\ 8\ users\ =\ 320\ patterns$.

Results

The in-air accuracy of Asterisk came out to be 95.0% (95% CI [86.3, 103.7]), while for Obelisk it was 81.2% (95% CI [71.3, 91.2]). The difference was statistically significant ($F(1,7) = 15.400$, $p < .01$, $\eta_p^2 = .688$). The mean durations were slightly higher (Asterisk: 11.4s, 95% CI [9.4, 13.4], Obelisk: 8.9s, 95% CI [8.0, 9.8]). Notice that the accuracy of in-air 45° Asterisk is almost equal to the on-surface accuracy of 45° 3cm Asterisk. Asterisk in air does not suffer from occlusion which might have helped in getting a comparable performance. Obelisk's performance, however, is not maintained. One reason is that even with the smaller patterns, user's hand still gets stretched out and it's difficult to judge and perform exact strokes.

With 1-digit error correction, the accuracies jumped to 98.7% for Asterisk and 96.9% for Obelisk, thus matching Obelisk 45°'s on-surface corrected performance. Further, the undetected error % were around 1% for both. This shows that most errors in Obelisk were single stroke errors. One big advantage of in-air stroking was that since the user's fingers were not resting on a surface, they rarely performed distal-phalange movements. Secondly, for Asterisk, the occlusion problem was almost non-existent.

Our in-air study did not handle all in-air scenarios including larger codes at a distance and on-display at various angles with respect to the user. However, it does indicate overcoming of a very crucial limitation: the requirement of physical contact with the surface of the code. *The user's performance will not be affected if instead of touching the surface they simply hover over it and perform the strokes.* This should be true even for 36° and 30° as long as the finger hovers right next to the surface. Larger distances, lower basis-angles and other in-air scenarios merit a separate exploration.

SUMMARY & IMPLICATIONS

We now summarize the takeaways from our evaluation: (1) Asterisk provides higher accuracy for higher information capacities than Obelisk. Considering 95% accuracy as the threshold, Asterisk 45° provides 16.8 million unique sequences (24 bits) in 10.3s and Asterisk 36°, 100k sequences (16.6 bits) in 7.5s. Obelisk is faster than Asterisk owing to its single stroke design, but only provides a maximum of 12k sequences (13.6 bits) for 95% accuracy in 5.5s. (2) Error correction takes Asterisk to a near-perfect accuracy but for only 20736 sequences (14.3 bits). A suitable middle-ground is error detection which leads to a <2% undetected error rate for Asterisk with 10 million sequences (23.3 bits) while requiring the user to redo the trace

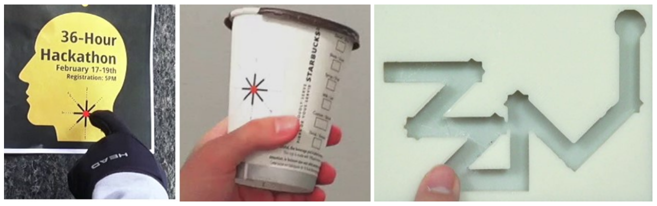


Figure 13. Application Scenarios: (left) Using with gloves. (mid) Use on portable objects. (right) 3D Printed grooved Obelisk

5% of the time. (3) Obelisk's accuracy for 36° and lower is too low for practical use. (4) Asterisk's accuracy does not go down with basis angle, and therefore smaller basis angles hold promise for both a fast and high information code. (5) Asterisk 45°'s in-air accuracy shows that users do not necessarily have to touch the surface, but can hover over it and scan the pattern without loss of accuracy. Obelisk 45°'s in-air accuracy is low, but improves to 96.9% with error correction. (6) Both Asterisk and Obelisk were agnostic to vertical vs. horizontal surface orientation (in addition to being agnostic of finger and ring orientation). This should hold true for slanted surfaces without loss of generality. (7) Longer lengths reduce deviations and therefore improve accuracy. The 1.5cm stroke length offers no speed advantage in addition to being significantly less accurate than 3cm for both Asterisk and Obelisk. (8) Since Obelisk will be less affected by the increased duration of a longer stroke, lengths >3cm may result in boosting its accuracy and merits investigation. (9) The stroke velocity vector faithfully reproduces the actual motion directions, but gets deviated due to overshooting and false starts. Asterisk is less prone to these deviations due to its double-stroke design. Future motion code designs should aim to minimize these issues while avoiding redundancy.

EXAMPLE SCENARIOS AND APPLICATIONS

Motion codes enable easy walk up & scan which can be useful in a variety of scenarios. For example, art pieces in a museum can be accompanied with motion codes that link to audio information that the user can listen to. In such a case, the user installs the museum audio app in their ring that recognizes the code upon tracing. Motion codes would be even more useful in museums (and other art installations) where cameras are not permitted and therefore optical QR codes are not an option. Such audio linking could be useful in various other scenarios, such as for product details in a retail store, for encoding audio labels in children's toys (for instance human anatomy model toys), or in books with supplemental audio.

Motion codes also support on-the-go usecases. For example, coffee cups can contain motion codes as part of a loyalty rewards program which the users can scan (Figure 13 (middle)). Since motion codes depend on the IMU readings, users can scan them without taking their gloves off, further boosting usability in on-the-go use (Figure 13 (left)).

Obelisk codes can be engraved into surfaces (Figure 13 (right)). Since Obelisk is self-guiding, they could potentially be scanned by visually impaired users who are unable to use optical codes. However, this requires a serious investigation into the self-guiding design of the code such that the accuracy of decoding is maintained.

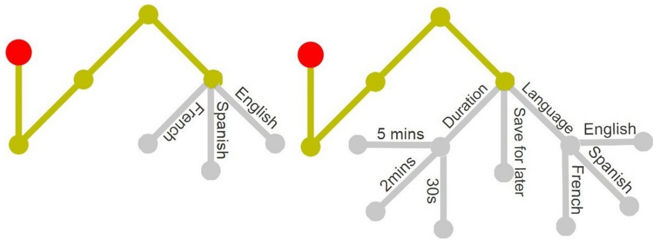


Figure 14. Motion codes with interactive actions. (left) User can trace this motion code which is linked to audio information and then choose their preferred language. (A motion code that enables multi-step actions.)

Since a motion code is already visible, it can be transformed into an optically scannable code by adding position, alignment and quiet zone markings to the pattern. This would result in a *hybrid optical-motion code* that can be scanned both optically and via motion. The users can then choose their preferred way of scanning. For instance, this provides the users an option to scan a distant code optically, and a proximate code via motion.

Motion Codes enable a unique form of interactivity where each line is an interactive step. This can be utilized to enable novel passive tags that enable multi-choice actions. For instance, users can scan an audio-linked tag and then choose their preferred language (Figure 14(left)). Another example is to use it for leaving quick product ratings, for example, after a restaurant meal, the user can scan the code and choose a rating from 1 to 5. Such interaction can be expanded to multiple steps (Figure 14 (right)) for richer options.

UTILITY, LIMITATIONS AND IMPROVEMENTS

Utility and comparison with QR codes

Most participants liked the simple walk-up-and-scan approach of the motion code. They liked that this did not require them to reach for their phone into the pocket. One participant indicated that they can hold bags in both hands and still wiggle-out the thumb, index finger to perform the scan which could be very handy. Some participants mentioned that it is a very useful functionality to have if they are already wearing smartrings, but they won't wear a smartring only for using this. Most participants indicated they would want durations to be short. However, opinions were split on the acceptable durations, from 2-3s to 7-8s. One participant compared it with a QR code, mentioning that as long as the duration is lower than the time it takes to get the phone out & scan, they would use it.

As we mentioned earlier, motion codes may be more pertinent than QR codes in certain contexts - places where cameras are not allowed, dark environments or for visually impaired users. Further, motion codes can function as hybrid optical-motion codes allowing users to choose their scanning method on the fly. Motion codes can also enable newer interactions such as scanning and invoking action in a single trace. However, as we discuss next, we also recognize that for mass adoption usecases, motion codes need to match or override QR code performance.

Limitations

Our current implementation of motion codes has a few limitations. The motion code only encodes the data without encoding the parameters of the encoding scheme such as basis angle and size. Such information may be needed if the decoding app needs to support motion codes with different parameters. However, this is easily solvable by encoding this information as part of the first line in the code. Secondly, our motion codes are not resistant to distal phalange movements and cannot handle smooth motion traces that are without momentary pauses between strokes. This problem is more difficult and requires more advanced signal processing algorithms. This is related to the issue of extracting reliable displacement data which can massively boost the data density. An IMU sensor with a higher sampling rate (BNO055 allows 100Hz) and position detection algorithms that can work well. Thirdly, button clicking adds an extra step. There are three ways to approach replacing the click with gestural delimiters: a unique start/stop pattern, tapping (double/triple), or implicit detection where the decoder continually runs and deems specific peak sequences to be valid motion code traces. This requires further exploration.

We have talked about IMU errors and user stroking errors. A third indirect source of error that affects both is *environmental error*, caused from uneven surfaces or scanning while moving. The current algorithm assumes the code to be on a smooth surface which is fixed. Uneven, rough surfaces will lead to lower accuracy. Similarly, scanning while walking or inside a moving subway introduces an uneven external acceleration which affects accuracy. Motion coding schemes and algorithms that are tolerant to these issues is an open problem for future work.

Finally, motion codes as a concept is not limited to Asterisk or Obelisk, to IMUs, or to finger wearables. We hope that our work encourages explorations into varied motion coding schemes that rely on different sensing solutions and explore other wearable devices such as smartwatches. Overall, we can summarize future work into three broad threads - *Technical:* improve information density, speed, sensing methods, and applicability for curved and in-air tracing; *Interaction possibilities:* hybrid codes, interactive actions etc.; *Utility:* for specific application areas or users (museums, visually impaired use).

CONCLUSION

We proposed motion codes that are passive, inexpensive tags that can be decoded using low-power, miniaturizable hardware that can be worn on the finger. We described the desired usability and functional properties of motion codes, based on which we design two motion code schemes, Asterisk and Obelisk. We implemented and then evaluated both against varying lengths, orientations, and information densities. We reported on the resulting accuracies and durations and investigated error detection and correction. We conducted an in-depth analysis of motion deviations. We further reported on an in-air motion code evaluation. Finally, we summarized our findings, discussed example applications, and limitations and improvements. We believe that motion codes is highly useful concept and our work traces the first line in their investigation.

REFERENCES

1. Adafruit. 2017. Adafruit BNO055 Absolute Orientation Sensor. (2017). https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/overview

2. Android Dev. 2017. Using Wrist Gestures | Android Developers. (2017). https://developer.android.com/training/wearables/ui/wrist-gestures.html

3. E Arendarenko. 2009. A study of comparing RFID and 2D barcode tag technologies for pervasive mobile applications. *Master, Dep. Comput. Sci.* (2009). https://www.researchgate.net/profile/Ernest

4. Daniel Ashbrook, Patrick Baudisch, and Sean White. 2011. Nenya: subtle and eyes-free mobile input with a magnetically-tracked finger ring. In *Proc. 2011 Annu. Conf. Hum. factors Comput. Syst. - CHI '11*. ACM Press, New York, New York, USA, 2043. DOI: http://dx.doi.org/10.1145/1978942.1979238

5. Gilles Bailly, Dong-Bach Vo, Eric Lecolinet, and Yves Guiard. 2011. Gesture-aware Remote Controls: Guidelines and Interaction Technique. In *Proceedings of the 13th International Conference on Multimodal Interfaces (ICMI '11)*. ACM, New York, NY, USA, 263–270. DOI: http://dx.doi.org/10.1145/2070481.2070530

6. Liwei Chan, Yi-Ling Chen, Chi-Hao Hsieh, Rong-Hao Liang, and Bing-Yu Chen. 2015. CyclopsRing: Enabling Whole-Hand and Context-Aware Interactions Through a Fisheye Ring. In *Proc. 28th Annu. ACM Symp. User Interface Softw. Technol. - UIST '15*. ACM Press, New York, New York, USA, 549–556. DOI: http://dx.doi.org/10.1145/2807442.2807450

7. Ke-Yu Chen, Kent Lyons, Sean White, and Shwetak Patel. 2013. uTrack: 3D input using two magnetic sensors. In *Proc. 26th Annu. ACM Symp. User interface Softw. Technol. - UIST '13*. ACM Press, New York, New York, USA, 237–244. DOI: http://dx.doi.org/10.1145/2501988.2502035

8. Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. 2015. Orbits: Gaze Interaction for Smart Watches Using Smooth Pursuit Eye Movements. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software &#38; Technology (UIST '15)*. ACM, New York, NY, USA, 457–466. DOI: http://dx.doi.org/10.1145/2807442.2807499

9. Aakar Gupta, Muhammed Anwar, and Ravin Balakrishnan. 2016a. Porous Interfaces for Small Screen Multitasking using Finger Identification. In *Proc. 29th Annu. Symp. User Interface Softw. Technol. - UIST '16*. ACM Press, New York, New York, USA, 145–156. DOI: http://dx.doi.org/10.1145/2984511.2984557

10. Aakar Gupta and Ravin Balakrishnan. 2016. DualKey: Miniature Screen Text Entry via Finger Identification. In *Proc. 2016 CHI Conf. Hum. Factors Comput. Syst. - CHI '16*. ACM Press, New York, New York, USA, 59–70. DOI: http://dx.doi.org/10.1145/2858036.2858052

11. Aakar Gupta, Antony Irudayaraj, Vimal Chandran, Goutham Palaniappan, Khai N. Truong, and Ravin Balakrishnan. 2016b. Haptic Learning of Semaphoric Finger Gestures. In *Proc. 29th Annu. Symp. User Interface Softw. Technol. - UIST '16*. ACM Press, New York, New York, USA, 219–226. DOI: http://dx.doi.org/10.1145/2984511.2984558

12. Chris Harrison and Scott E. Hudson. 2009. Abracadabra. In *Proc. 22nd Annu. ACM Symp. User interface Softw. Technol. - UIST '09*. ACM Press, New York, New York, USA, 121. DOI: http://dx.doi.org/10.1145/1622176.1622199

13. Chris Harrison, Robert Xiao, and Scott Hudson. 2012. Acoustic barcodes. In *Proc. 25th Annu. ACM Symp. User interface Softw. Technol. - UIST '12*. ACM Press, New York, New York, USA, 563. DOI: http://dx.doi.org/10.1145/2380116.2380187

14. H Hu, J Tang, H Zhong, Z Xi, C Chen, and Q Chen. 2013. Invisible photonic printing: computer designing graphics, UV printing and shown by a magnetic field. *Sci. Rep.* (2013). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3601367/

15. Wolf Kienzle and Ken Hinckley. 2014. LightRing: always-available 2D input on any surface. In *Proc. 27th Annu. ACM Symp. User interface Softw. Technol. - UIST '14*. ACM Press, New York, New York, USA, 157–160. DOI:http://dx.doi.org/10.1145/2642918.2647376

16. Dingzeyu Li, Avinash S. Nair, Shree K. Nayar, and Changxi Zheng. 2017. AirCode: Unobtrusive Physical Tags for Digital Fabrication. (jul 2017). DOI: http://dx.doi.org/10.1145/3126594.3126635

17. Todd K. Moon. 2005. *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience.

18. Suranga Nanayakkara, Roy Shilkrot, and Pattie Maes. 2012. EyeRing: an eye on a finger. In *Proc. 2012 ACM Annu. Conf. Ext. Abstr. Hum. Factors Comput. Syst. Ext. Abstr. - CHI EA '12*. ACM Press, New York, New York, USA, 1961. DOI: http://dx.doi.org/10.1145/2212776.2223736

19. Anh Nguyen and Amy Banic. 2015. 3DTouch: A wearable 3D input device for 3D applications. In *2015 IEEE Virtual Real.* IEEE, 55–61. DOI: http://dx.doi.org/10.1109/VR.2015.7223324

20. Ian Oakley and Junseok Park. 2009. Motion Marking Menus: An Eyes-free Approach to Motion Input for Handheld Devices. *Int. J. Hum.-Comput. Stud.* 67, 6 (June 2009), 515–532. DOI: http://dx.doi.org/10.1016/j.ijhcs.2009.02.002

21. Masa Ogata, Yuta Sugiura, Hirotaka Osawa, and Michita Imai. 2012. iRing: : intelligent ring using infrared reflection. In *Proc. 25th Annu. ACM Symp. User interface Softw. Technol. - UIST '12*. ACM Press, New York, New York, USA, 131. DOI: http://dx.doi.org/10.1145/2380116.2380135

22. Melania Susi, Valérie Renaudin, and Gérard Lachapelle. 2013. Motion mode recognition and step detection algorithms for mobile phone users. *Sensors (Basel).* 13, 2 (jan 2013), 1539–62. DOI: http://dx.doi.org/10.3390/s130201539

23. Kevin Townsend. 2016. Adafruit Unified BNO055 Driver. (2016). https://github.com/adafruit/Adafruit

24. Hsin-Ruey Tsai, Min-Chieh Hsiu, Jui-Chun Hsiao, Lee-Ting Huang, Mike Chen, and Yi-Ping Hung. 2016. TouchRing: subtle and always-available input using a multi-touch ring. In *Proc. 18th Int. Conf. Human-Computer Interact. with Mob. Devices Serv. Adjun. - MobileHCI '16*. ACM Press, New York, New York, USA, 891–898. DOI: http://dx.doi.org/10.1145/2957265.2961860

25. Koji Tsukada and Michiaki Yasumura. 2001. Ubi-finger: Gesture input device for mobile use. In *Ubicomp 2001 Informal Companion Proceedings*. 11.

26. Eduardo Velloso, Markus Wirth, Christian Weichel, Augusto Esteves, and Hans Gellersen. 2016. AmbiGaze: Direct Control of Ambient Devices by Gaze. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16)*. ACM, New York, NY, USA, 812–817. DOI: http://dx.doi.org/10.1145/2901790.2901867

27. David Verweij, Augusto Esteves, Vassilis-Javed Khan, and Saskia Bakker. 2017. WaveTrace: Motion Matching Input Using Wrist-Worn Motion Sensors. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '17)*. ACM, New York, NY, USA, 2180–2186. DOI: http://dx.doi.org/10.1145/3027063.3053161

28. Mélodie Vidal, Andreas Bulling, and Hans Gellersen. 2013. Pursuits: Spontaneous Interaction with Displays Based on Smooth Pursuit Eye Movement and Moving Targets. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '13)*. ACM, New York, NY, USA, 439–448. DOI:http://dx.doi.org/10.1145/2493432.2493477

29. Karl D. D. Willis and Andrew D. Wilson. 2013. InfraStructs. *ACM Trans. Graph.* 32, 4 (jul 2013), 1. DOI: http://dx.doi.org/10.1145/2461912.2461936

30. Chao Xu, Parth H. Pathak, and Prasant Mohapatra. 2015. Finger-writing with Smartwatch. In *Proc. 16th Int. Work. Mob. Comput. Syst. Appl. - HotMobile '15*. ACM Press, New York, New York, USA, 9–14. DOI: http://dx.doi.org/10.1145/2699343.2699350

31. Xing-Dong Yang, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. 2012. Magic finger: always-available input through finger instrumentation. In *Proc. 25th Annu. ACM Symp. User interface Softw. Technol. - UIST '12*. ACM Press, New York, New York, USA, 147. DOI: http://dx.doi.org/10.1145/2380116.2380137